**Linked List**

**1. 개념**

**< 배열(Array)와 리스트(List) >**

링크드 리스트 이전에 먼저 배열과 리스트의 개념을 짚고 넘어가야한다.

**\* 배열**

: 메모리상에서 연속적인 공간을 할당받아 데이터를 저장한다. 배열은 인덱스로 원소에 접근하는데, 배열에서의 인덱스는 원소의 주소를 가리키는 식별자이다.

이를 통한 장점은 아주 빠르게 특정 원소 접근가능!

**\* 리스트**

: 메모리 상, 연속적 공간에 할당되지 않을 수 있다. 즉, 첫번째 원소 주소를 알아도 단순히 다음 원소 주소를 알 수 없다. 그래서 리스트는 다음 원소를 가리키는 ‘포인터’를 사용하여 각 원소의 순서를 구현한다.

이를 통한 장점은 배열보다 데이터의 삽입/삭제가 빠르다. 앞/뒤 원소의 포인터만 바꿔주면 되기 때문. 하지만 인덱스를 사용하여 특정원소에 직접접근이 불가능하여 원소를 검색하기 위해선 첫번째 원소부터 “선형 검색” 해야하는 단점!

***\*\*\* 주의 \*\*\****

**\* 파이썬에서의 리스트**

파이썬에서 리스트는 위의 컴퓨터 공학적 의미와 다르다. 실제 메모리 공간에 모든 원소를 연속적으로 배치하는 ‘배열’로서 구현되어 있어서, 인덱스로 원소 접근이 가능하다. 그리고 실제 배열과 다르게 길이가 고정되어있지 않아, 데이터의 추가/삭제가 편리하다. (이는 파이썬 리스트에서 실제 필요한 메모리 보다 조금 더 여유있게 메모리를 마련해놓기 때문)

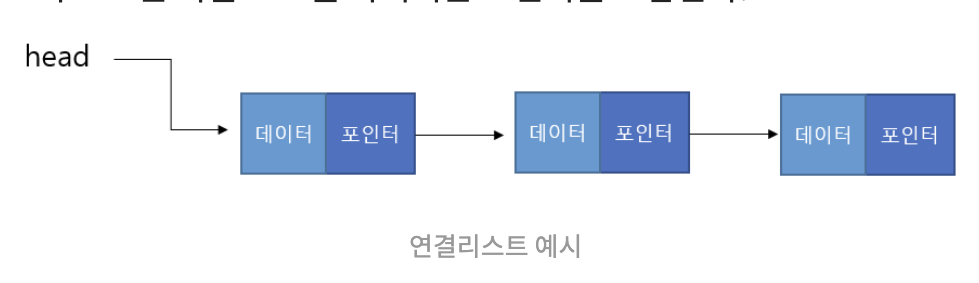
**< 링크드 리스트 >**

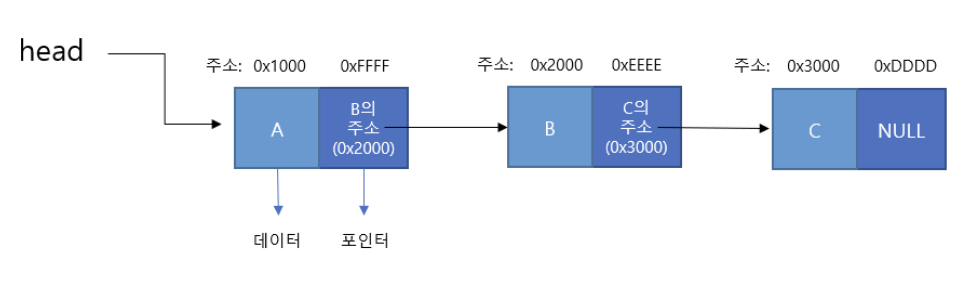
- 리스트는 순서가 있는 데이터를 늘어놓은 형태의 자료구조, 각 원소는 노드라고 하며 각 노드는 데이터와 그 다음 노드를 가리키는 포인터를 갖고 있다.

- 링크드 리스트란 이 노드들을 연결시킨 자료구조

- 맨 앞과 맨 끝 노드를 각각 머리(HEAD), 꼬리(Tail)노드라고 부르며 본인의 앞쪽 노드를 ‘전임노드’, 뒤쪽 노드는 ‘후임노드’라고 부른다.

- 각 노드의 포인터 변수는 다음 노드의 데이터의 주소를 값으로 가진다. 또한 각 포인터 변수의 주소도 따로 존재한다.





**2. 동작과정**

**새로운 노드 생성 시)**

: newnode 생성 후, head가 기존에 가리키는 노드데이터를 newnode 포인터 또한 가리키게 한다. 그리고 head가 newnode를 가리키면 끝.

: 중간에 노드 생성시에는 head 대신 전임노드를 기준으로 같은 방법을 이용

**3. 예제 코드**

“Reverse Linked List” 총 3가지 방법 풀이